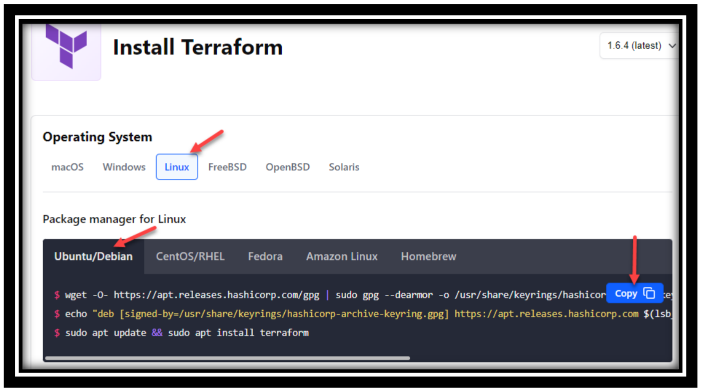
Working with Terraform

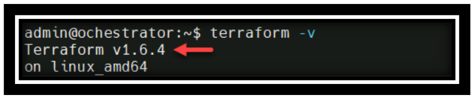
Task – 1: Simple Terraform Example

1. Follow the instructions from https://developer.hashicorp.com/terraform/install to Install Terraform on the Ubuntu orchestrator.



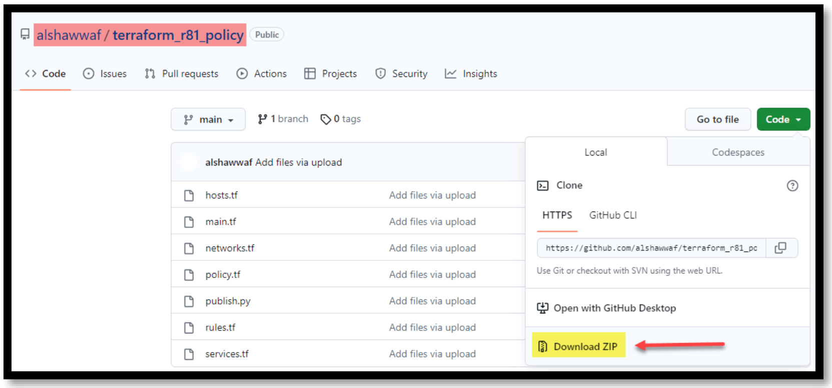
```
$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicor
$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb
$ sudo apt update && sudo apt install terraform
```

2. Run the command terraform -v to see the current installed version of Terraform.



```
admin@ochestrator:~$ terraform -v
Terraform v1.6.4
on linux_amd64
```

3. Download the project files from GitHub https://github.com/alshawwaf/terraform_r81_policy and extract the files.



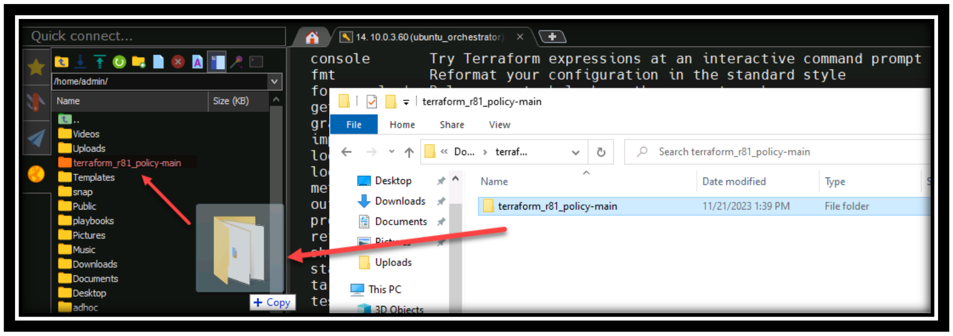
alshawwaf Add files via upload

hosts.tf Add files via upload
main.tf Add files via upload
networks.tf Add files via upload
policy.tf Add files via upload
publish.py Add files via upload
rules.tf Add files via upload
services.tf Add files via upload

https://github.com/alshawwaf/terraform_r81_pc

Use Git or checkout with SVN using the web URL.

4. Copy the folder we downloaded to the Ubuntu Orchestrator.

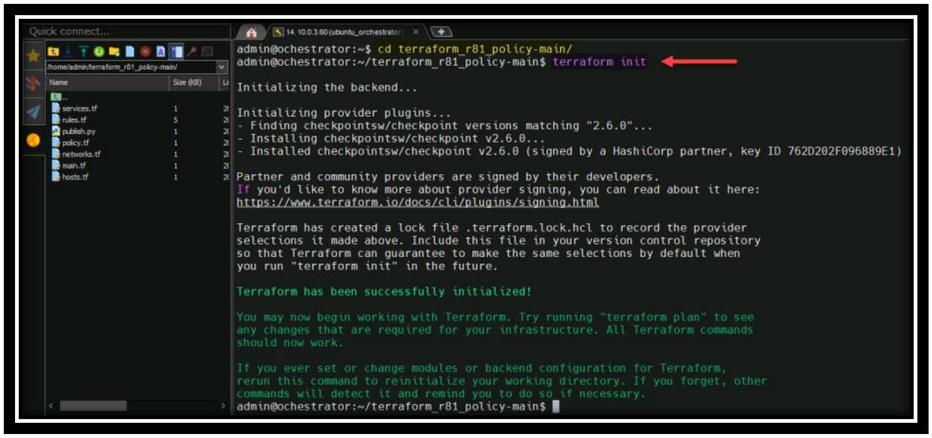


```
console        Try Terraform expressions at an interactive command prompt
fmt            Reformat your configuration in the standard style
```

5. Review the configurations files in the terraform directory.

6. Run the following command to initialize Terraform and the Check Point POC provider for Terraform. This will download the latest Check Point Terraform provider from Hashicorp.

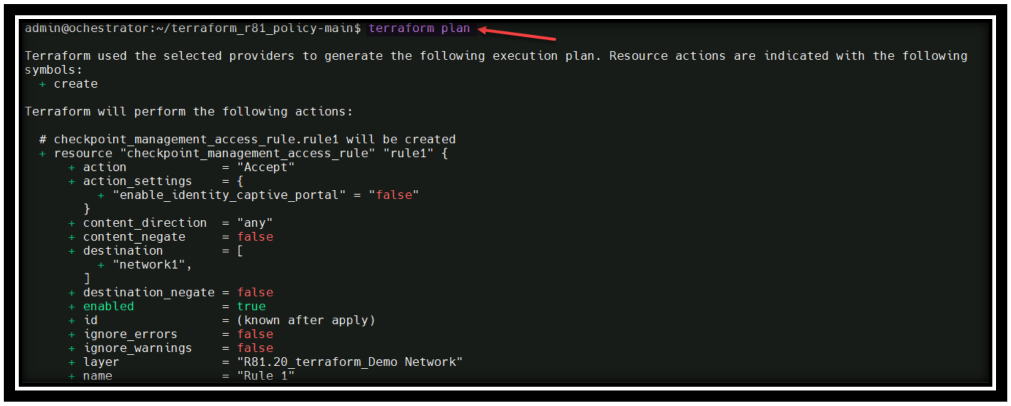
```
terraform init
```



```
admin@ochestrator:~$ cd terraform_r81_policy-main/
admin@ochestrator:~/terraform_r81_policy-main$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding checkpointsw/checkpoint versions matching "2.6.0"...
- Installing checkpointsw/checkpoint v2.6.0...
- Installed checkpointsw/checkpoint v2.6.0 (signed by a HashiCorp partner, key ID 762D202F096889E1)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
admin@ochestrator:~/terraform_r81_policy-main$
```

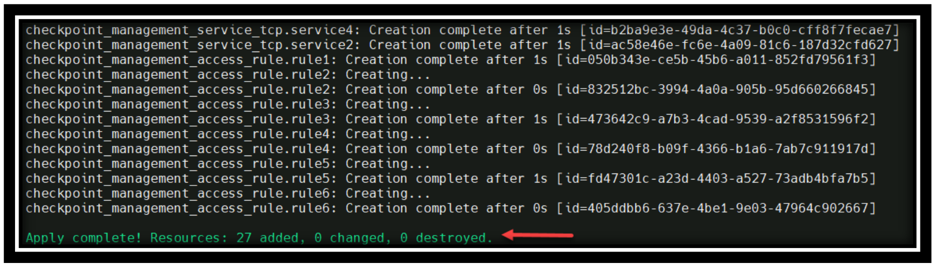
7. Next, we will run a plan to see what would actually be done to build the objects that are described in the Simple Example directory.

```
terraform plan
```



```
admin@ochestrator:~/terraform_r81_policy-main$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # checkpoint_management_access_rule.rule1 will be created
  + resource "checkpoint_management_access_rule" "rule1" {
      + action              = "Accept"
      + action_settings     = {
          + "enable_identity_captive_portal" = "false"
        }
      + content_direction   = "any"
      + content_negate      = false
      + destination         = [
          + "network1",
        ]
      + destination_negate = false
      + enabled             = true
      + id                  = (known after apply)
      + ignore_errors       = false
      + ignore_warnings     = false
      + layer               = "R81.20_terraform_Demo Network"
      + name                = "Rule 1"
```

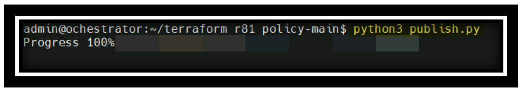
8. How many objects does terraform need to add, how many need deleted?
9. Now we can apply our configuration. Run the following command:
   `terraform apply`



```
checkpoint_management_service_tcp.service4: Creation complete after 1s [id=b2ba9e3e-49da-4c37-b0c0-cff8f7fecae7]
checkpoint_management_service_tcp.service2: Creation complete after 1s [id=ac58e46e-fc6e-4a09-81c6-187d32cfd627]
checkpoint_management_access_rule.rule1: Creation complete after 1s [id=050b343e-ce5b-45b6-a011-852fd79561f3]
checkpoint_management_access_rule.rule2: Creating...
checkpoint_management_access_rule.rule2: Creation complete after 0s [id=832512bc-3994-4a0a-905b-95d660266845]
checkpoint_management_access_rule.rule3: Creating...
checkpoint_management_access_rule.rule3: Creation complete after 1s [id=473642c9-a7b3-4cad-9539-a2f8531596f2]
checkpoint_management_access_rule.rule4: Creating...
checkpoint_management_access_rule.rule4: Creation complete after 0s [id=78d240f8-b09f-4366-b1a6-7ab7c911917d]
checkpoint_management_access_rule.rule5: Creating...
checkpoint_management_access_rule.rule5: Creation complete after 1s [id=fd47301c-a23d-4403-a527-73adb4bfa7b5]
checkpoint_management_access_rule.rule6: Creating...
checkpoint_management_access_rule.rule6: Creation complete after 0s [id=405ddbb6-637e-4be1-9e03-47964c902667]

Apply complete! Resources: 27 added, 0 changed, 0 destroyed.
```

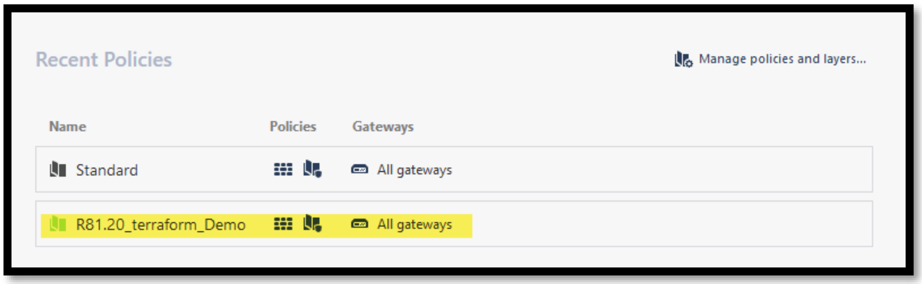
10. Once the apply is complete run the follow to publish the changes with the python script

    `python3 publish.py`



```
admin@ochestrator:~/terraform r81 policy-main$ python3 publish.py
Progress 100%
```

11. Verify that the objects are in the .tf files were created in the R81 SmartConsole



Recent Policies

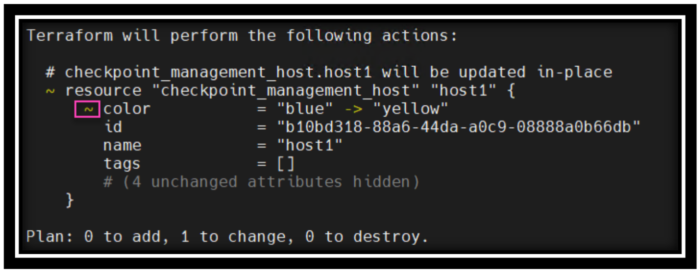
Manage policies and layers...

Name	Policies	Gateways
Standard		All gateways
R81.20_terraform_Demo		All gateways

Task – 2: Update the configuration

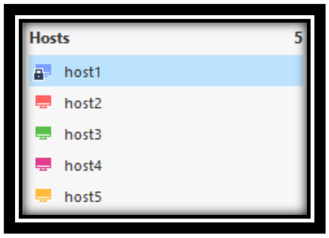
Now that we have built some elements of our infrastructure, we can see how easy it is to apply new changes to the configuration by simply changing the .tf files.

1. Change the color of a host in the hosts.tf file and run the terraform plan and terraform apply. What is going to be changed?



```
Terraform will perform the following actions:

  # checkpoint_management_host.host1 will be updated in-place
  ~ resource "checkpoint_management_host" "host1" {
      ~ color             = "blue" -> "yellow"
        id                = "b10bd318-88a6-44da-a0c9-08888a0b66db"
        name              = "host1"
        tags              = []
        # (4 unchanged attributes hidden)
    }

Plan: 0 to add, 1 to change, 0 to destroy.
```

2. Check to see if that host is now locked in SmartConsole



3. Publish the changes and verify again if it is updated in SmartConsole.

4. Next add 2 new hosts in the hosts.tf file and add a new network

5. Now apply this new configuration again using what you have learned so far.

6. Add an additional rule with the new network and new hosts

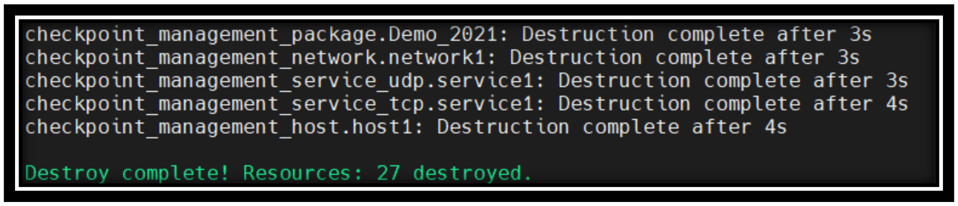
7. Now use SmartConsole to disable a rule, negate a source and make a rule ANY/ANY/ANY. Publish your changes

8. Run a terraform apply and verify what is going to be updated/fixed.

9. Did the policy go back to where it started based on what is in the terraform files?

10. Now delete the entire configuration using the following command

```
terraform destroy
```



```
checkpoint_management_package.Demo_2021: Destruction complete after 3s
checkpoint_management_network.network1: Destruction complete after 3s
checkpoint_management_service_udp.service1: Destruction complete after 3s
checkpoint_management_service_tcp.service1: Destruction complete after 4s
checkpoint_management_host.host1: Destruction complete after 4s

Destroy complete! Resources: 27 destroyed.
```

11. Once the destroy is complete run the follow to publish the changes with the python script

```
python3 publish.py
```

End Of Lab